AFRL-IF-RS-TR-2002-147
Final Technical Report
June 2002

A MATLAB COMPILATION ENVIRONMENT FOR ADAPTIVE COMPUTING SYSTEMS

Northwestern University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. D002

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

APPROVED: Martin J Walter

MARTIN WALTER
Project Engineer

FOR THE DIRECTOR: [illegible]

MICHAEL L. TALBERT, Technical Advisor
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** JUNE 2002	**3. REPORT TYPE AND DATES COVERED** Final Mar 98 – Aug 01

4. TITLE AND SUBTITLE A MATLAB COMPILATION ENVIRONMENT FOR ADAPTIVE COMPUTING SYSTEMS	**5. FUNDING NUMBERS** C - F30602-98-2-0144, PE - 62301E, PR - D002, TA - TC, WU - 01
6. AUTHOR(S) Prithviraj Banerjee	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Northwestern University, Center for Parallel and Distributed Computing, 2145 Sheridan Road, Evanston IL 60208	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency, 3701 North Fairfax Drive, Arlington Virginia 22203-1714; AFRL/IFTC, 26 Electronic Parkway, Rome New York 13441-4514	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2002-147

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Martin Walter/IFTC/(315) 330-4102/ Martin.Walter@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

This report provides a brief summary of the research and development of a compiler for a mix of general purpose processors and adaptive computing processors from MATLAB. It incorporates a list of publications resulting from this research.

14. SUBJECT TERMS MATLAB, Compilation, Adaptive Computing Systems	**15. NUMBER OF PAGES** 20
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

TABLE OF FIGURES

SUMMARY

This final report summarizes the research results obtained during the MATCH compiler project on “A MATLAB Compilation Environment for Adaptive Computing Systems,” supported at Northwestern University between March 1998 to August 2001. The objective of the MATCH project was to make it easier for DOD users to develop efficient codes for adaptive computing systems. We have developed a compiler that takes in DOD applications written in a high-level language (MATLAB) and generates efficient low level code that runs on a distributed environment of commercial-off-the-shelf (COTS) FPGAs, embedded processors, and digital signal processors. The main features of the compiler are:

1. It enable the users to reduce the code development times for adaptive applications from weeks using manual approaches to hours using compiler tools.
2. It produce efficient codes that are within a factor of 2-4 of the best manual approach with respect to optimizing resources under performance constraints, or optimizing performance under resource constraints.

The project URL is at: http://www.ece.nwu.edu/cpdc/Match/Match.html
The results of the MATCH compiler have been transferred to a startup company called AccelChip, Inc. (formerly called MACH DESIGN SYSTEMS). The company was founded by two of the PIs of the proposal, Prith Banerjee and Alok Choudhary, and two of the Ph.D. students, Malay Haldar and Anshuman Nayak.

1. INTRODUCTION

Efficient high-level design tools that can map behavioral descriptions of signal and image processing applications to FPGA architectures are one of the key requirements to fully leverage FPGAs for high-throughput computations and meet time to market pressures. Currently, most FPGA designs are entered at the level of Register Transfer Level (RTL) VHDL or Verilog. It is widely recognized that there is a need for design tools at the high level using languages such as C/C++ or MATLAB. MATLAB is an extremely popular language in the signal and image processing community with over 500,000 users. A direct synthesis path from MATLAB into hardware would be very useful. The MATCH compiler at Northwestern University takes as input algorithms described in MATLAB, and generates Register Transfer Level (RTL) VHDL. The RTL VHDL then can be mapped to FPGAs using commercial tools. The input application is mapped to multiple FPGAs by parallelizing the application and embedding computation and synchronization primitives automatically. Our compiler infers the minimum number of bits required to represent the variables through a precision inferencing analysis framework. The compiler can leverage optimized Intellectual Property (IP) cores to enhance the hardware generated. The compiler also exploits parallelism in the input algorithm by pipelining in the presence of resource constraints. We have demonstrated the utility of the compiler by synthesizing hardware for a couple of signal/image processing algorithms and comparing them to manually designed hardware.

2. MODELS, ASSUMPTIONS AND PROCEDURES

The MATCH project consisted of six research tasks. A brief description of each of the tasks is given below.

■Task 1: Development of a Testbed

- Development of a hardware testbed consisting of VME chassis, Motorola embedded boards, Transtech DSP boards, Annapolis Wildchild board, Annapolis WILDSTAR board, and FORCE 5V
- Acquired, installed and integrated hardware and software components together
- Performed measurements of basic operations on testbed

■Task 2: Implementation of Basic MATLAB Compiler

- Implementation of a compiler that takes in MATLAB and generates C code for the embedded and DSP processors, and RTL VHDL code for the FPGA board, and use commercial C and VHDL compilers to generate the object code
- Implemented MATLAB parser, AST builder, basic MATLAB to C compiler with library approach
- Implemented a basic compiler to convert MATLAB into C automatically
- Implemented a dynamic runtime system called MATCH Virtual Machine
- Implemented some basic type and shape inferencing for MATLAB variables
- Designed and implemented an algorithm to map MATLAB to RTL VHDL for FPGAs for single FPGA
- Designed and implemented an algorithm to map MATLAB to RTL VHDL for multiple FPGAs
- Evaluated compiler output with hand-mapped designs

■Task 3: Automatic Parallelism and Mapping

- Developed heuristics based on mixed integer linear programming to map a given dataflow graph of a MATLAB program on heterogeneous resources
  - Optimizing resources under performance constraints
  - Optimizing performance under resource constraints

•Evaluated the heuristics on various benchmarks (STAP, MPEG decoder, and about 100 synthetic benchmarks)

■Task 4: MATLAB Compiler Directives

- Developed directives to specify type, shape, size, precision, data distribution and alignment, task mapping, resource and time constraints
- Implemented directives within parser, stored information in AST for compiler to use

- Developed a report specifying the various directives

■Task 5: Evaluation of Adaptive Applications

- Evaluating applications such as Space Time Adaptive Processing, Hyperspectral Image Processing, and Honeywell benchmarks on MATCH testbed by writing applications in MATLAB and manually converting to the various components of testbed in C and VHDL and measuring performance
- Used to identify MATLAB functions

■Task 6: Library Development

- Identification and Implementations of Basic Primitives
- Implemented following functions on embedded, DSP boards: Matrix multiplication, Matrix addition, 1 and 2 D FFT
- Implemented following functions on Wildchild FPGA board: Vector Sum, Matrix multiplication, matrix addition, filtering, 2 D FFT, Complex matrix multiplication, wavelet transform
- Developed C Program Interfaces to all libraries from host to DSP and FPGA boards
- Developed interfaces to Integrated Sensors Inc. RTExpress MATLAB libraries
- Characterizing performance for varying problem size, varying number of processors and FPGAs, will be used by compiler

3. RESULTS AND DISCUSSION

We will now report on our results of our research under the six tasks in the MATCH project.

Testbed (Task 1): We have developed a hardware testbed of an adaptive computing system. The testbed consists of: (1) A Wildchild board from Annapolis Micro Systems containing 9 XILINX XC4010 FPGAs and 2 MB of memory; A Wildstar board from Annapolis Microsystems containing three Xilinx Virtex FPGAs, and 4 MB of memory; (3) Two Motorola MVME-2604 embedded boards containing 200 MHz PowerPC 604 microprocessor, 64 MB RAM; (4) One Transtech TDMB 428 DSP board consisting of four Texas Instruments 60 MHz TMS 320C40 digital signal processors; (5) A Force 5V board consisting of a microSPARC-II CPU and 64 MB of RAM. These boards are mounted on a VME bus chassis. We have integrated the various software components to have these various boards to operate together.

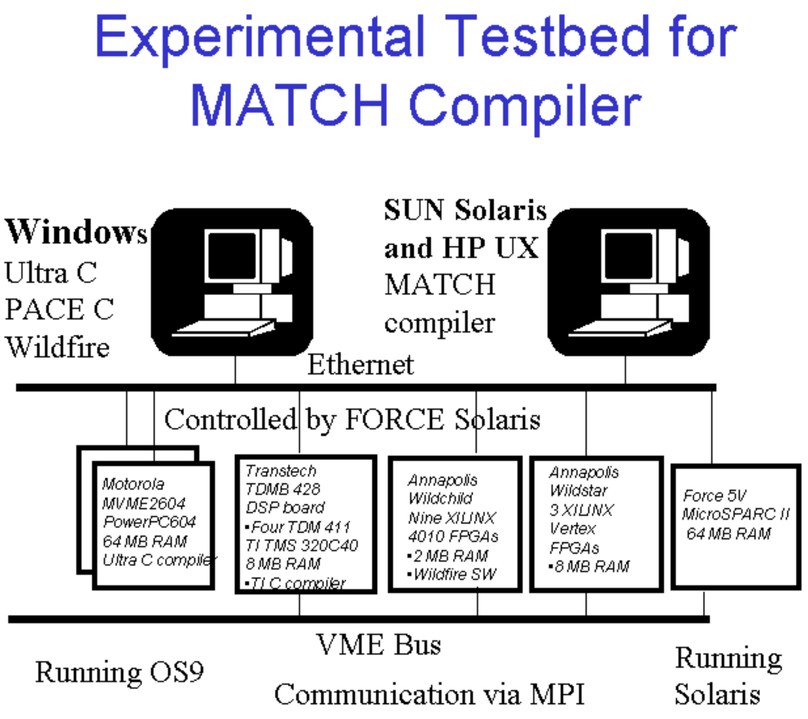

Experimental Testbed for MATCH Compiler
Windows
Ultra C
PACE C
Wildfire
SUN Solaris and HP UX
MATCH compiler
Ethernet
Controlled by FORCE Solaris
Motorola MVME2604 PowerPC604 64 MB RAM Ultra C compiler
Transtech TDMB 428 DSP board •Four TDM 411 TI TMS 320C40 8 MB RAM •TI C compiler
Annapolis Wildchild Nine XILINX 4010 FPGAs •2 MB RAM •Wildfire SW
Annapolis Wildstar 3 XILINX Vertex FPGAs •8 MB RAM
Force 5V MicroSPARC II 64 MB RAM
Running OS9
VME Bus
Communication via MPI
Running Solaris

Figure 1: Overview of the MATCH testbed.

Basic Compiler (Task 2): We have developed a MATCH compiler that takes MATLAB programs as input, and produces C programs to be mapped onto the embedded processors, and DSP processors, and RTL VHDL that will be mapped onto the FPGAs. In addition the compiler has the capability of making calls to library functions that are available on various targets. An overview of the MATCH compiler is shown in the figure below.

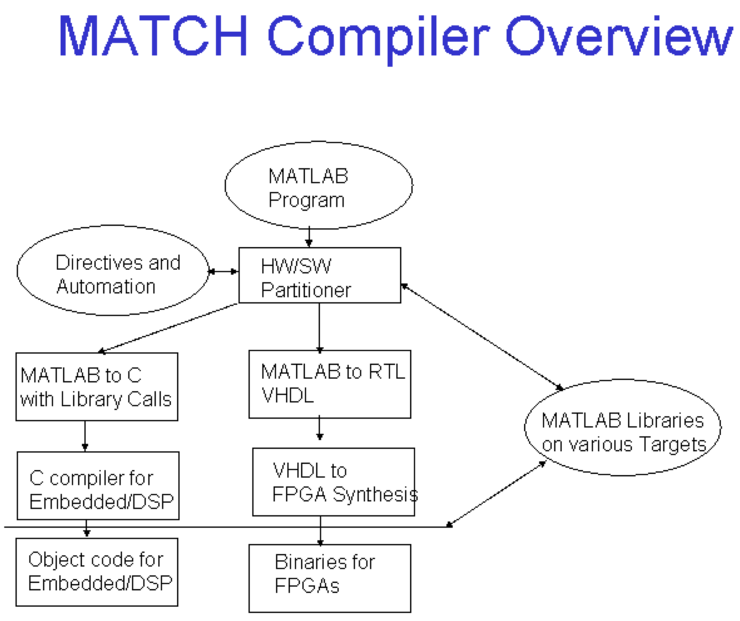

MATCH Compiler Overview
MATLAB Program
Directives and Automation
HW/SW Partitioner
MATLAB to C with Library Calls
MATLAB to RTL VHDL
MATLAB Libraries on various Targets
C compiler for Embedded/DSP
VHDL to FPGA Synthesis
Object code for Embedded/DSP
Binaries for FPGAs

Figure 2. Overview of the MATCH Compiler.

As part of the compiler effort we have developed a MATLAB to VHDL compiler which consists of the following steps. The front-end parses the input MATLAB program and builds a MATLAB AST (Abstract Syntax Tree). The input code may contain directives regarding the types, shapes and precision of arrays that cannot be inferred, which are attached to the AST nodes as annotations. This is followed by a type-shape inference phase. MATLAB variables have no notion of type or shape. The type-shape phase analyzes the input program to infer the type and shape of the variables present for which type/shape is not provided by directives. This is followed by a scalarization phase where the operations on matrices are expanded out into loops. In case optimized library functions are available for a particular operation, it is not scalarized and the IP core corresponding to the library function is used instead. The scalarized code is then passed through the parallelization phase. The parallelization phase attempts to exploit coarse grain parallelism by either splitting a loop onto multiple FPGAs on the board (data-parallel approach) or by putting different tasks onto different FPGAs and pipelining the output of one to the input of another (systolic approach). The parallelization phase relies on communication libraries implemented for the target architecture board to communicate between the different FPGAs. A state machine description in VHDL is then synthesized from the parallelized scalarized MATLAB code for each of the FPGAs. Most of the hardware related optimizations are performed on the VHDL AST. A precision inference scheme finds the minimum number of bits required to represent each variable in the AST. The precision information is used in instantiating customized IP blocks corresponding to the functions and operators. Transformations are then performed on the AST to optimize it according to the memory accesses present in the program and characteristics of the external memory. This is followed by a phase to perform optimizations like pipelining under resource constraints that alter parts of the state machine that was constructed earlier. Finally a traversal of the optimized VHDL AST produces the output code.

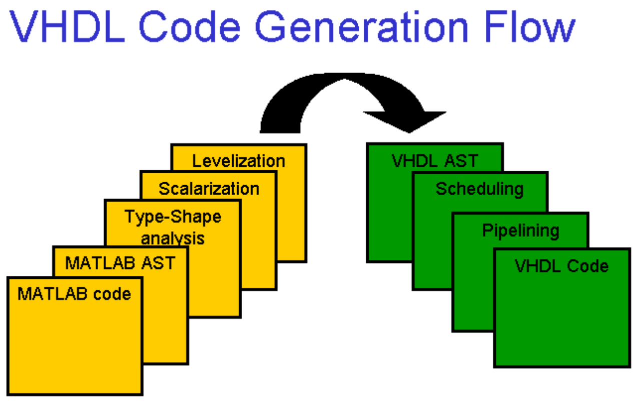


Figure 3. An overview of the compilation flow from MATLAB to VHDL.

We have developed a lot of optimizations as part of the compiler:

- Scheduling and allocation
- Pipelining of loops
- Precision analysis of variables
- Error analysis of variables
- Memory packing
- Intellectual property core integration

Some experimental results of the MATCH compiler are shown in Figure 3. Each manual design took at least a month to design. The MATCH compiler designed in less than a minute. In addition, the results of the MATCH compiler are equivalent (and sometimes better) to the manual design. The vertical scale shows the execution time of the compiler-generated code normalized to the manual design. The first bar shows the execution time for a manual design. The second bar shows the time with a basic compiler generating VHDL code without optimizations. The third one shows the time with the compiler with the pipelining optimization. It can be seen from these results that the results are comparable to manual designs.

- MATLAB programs compiled by MACH into RTL VHDL
- Synthesized by Synplify and Alliance for Xilinx 4028 FPGA
- Executed on Annapolis Wildchild board

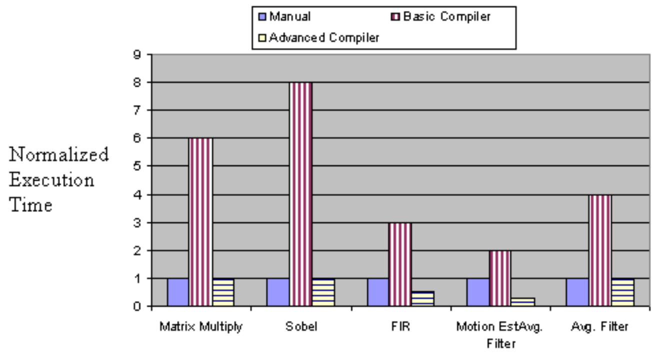
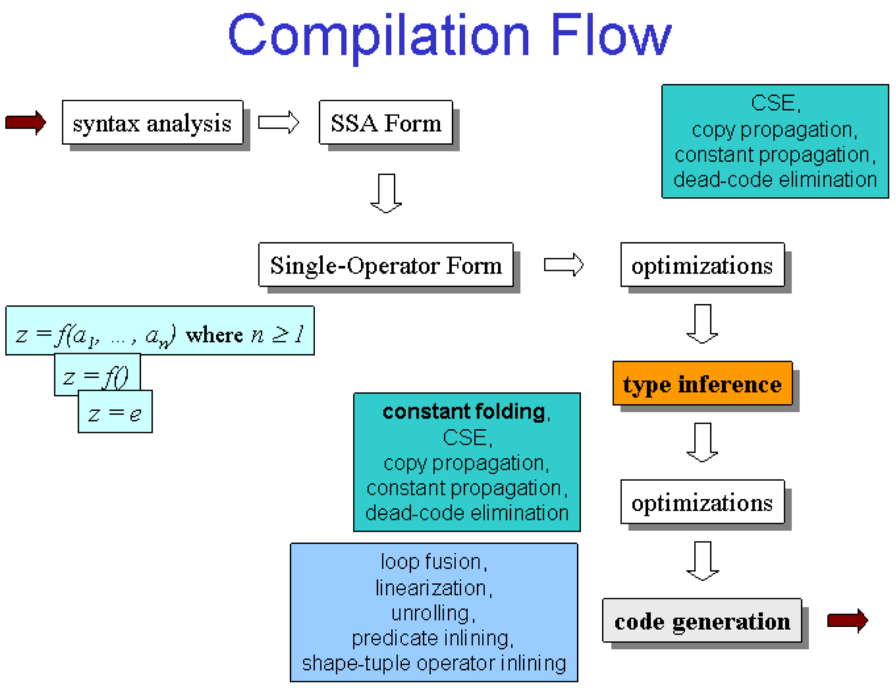


Figure 4. Results of our MATCH to VHDL compiler on five MATLAB benchmark programs.

Another part of the MATCH compiler was the compilation of MATLAB programs to C programs. A key part of this compiler was the automatic types and shapes of arrays in MATLAB. The various steps of this compiler are shown in Figure 5.



Compilation Flow
syntax analysis
SSA Form
CSE, copy propagation, constant propagation, dead-code elimination
Single-Operator Form
optimizations
$z = f(a_1, \ldots, a_n)$ where $n \geq 1$
$z = f()$
$z = e$
type inference
constant folding, CSE, copy propagation, constant propagation, dead-code elimination
optimizations
loop fusion, linearization, unrolling, predicate inlining, shape-tuple operator inlining
code generation

Figure 5. The various steps of a MATLAB to C compiler.

The first step is to parse the MATLAB program into the MATLAB AST. The AST is next converted into a Static Single Assignment (SSA) form and then into a Single Operator Form. Various optimizations such as common subexpression elimination, copy propagation, and dead-code elimination are performed. The next key stage is automatic determination of types and shapes of arrays. Finally there is code generation into C code using scalarization of vector statements into for loops. Numerous optimizations such as loop fusion, linearization, unrolling, are used.

Automatic Mapping (Task 3): We have developed automatic algorithms for partitioning and mapping the MATLAB programs on the heterogeneous target. We have developed algorithms for pipelining, partitioning, allocation of resources, and scheduling of the operations on the various platforms to perform time-constrained resource optimizations. We have developed a tool called SYMPHANY for performing the task of automated program partitioning and pipelining. Given a high-level sequential specification of the real-time computation with associated timing constraints (latency and throughput), the tool automatically arrives at a cost-effective solution to the system design problem using embedded processors, DSP processors, FPGAs. Our algorithm is based on a mixed integer linear programming formulation and uses an off-the-shelf LP solver called "lp_solve". We have applied our tool to the data flow graphs of three synthetic benchmarks and to the graphs for the STAP application and an MPEG decoder. In each benchmark, we have studied the solution to the problem for various combinations of throughputs and latency constraints. In each case the SYMPHANY tool gave the right solutions in terms of the number of pipeline stages used. It gave better solutions than a hand-optimized solution in most cases by about 10-20% in terms of the cost of the solution in dollars. An example set of results on the STAP application is shown below.

Improvement over Heuristic

- A generic heuristic algorithm is used as the baseline.
- Average 20-50% cost reduction.
- 70% cost reduction achieved in many cases.

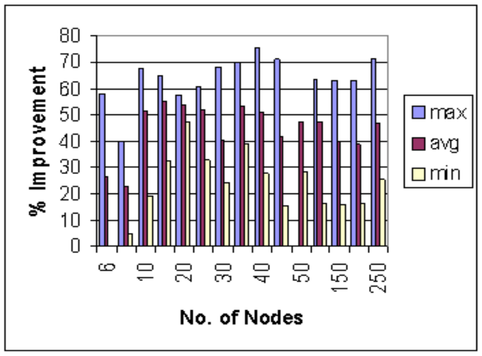
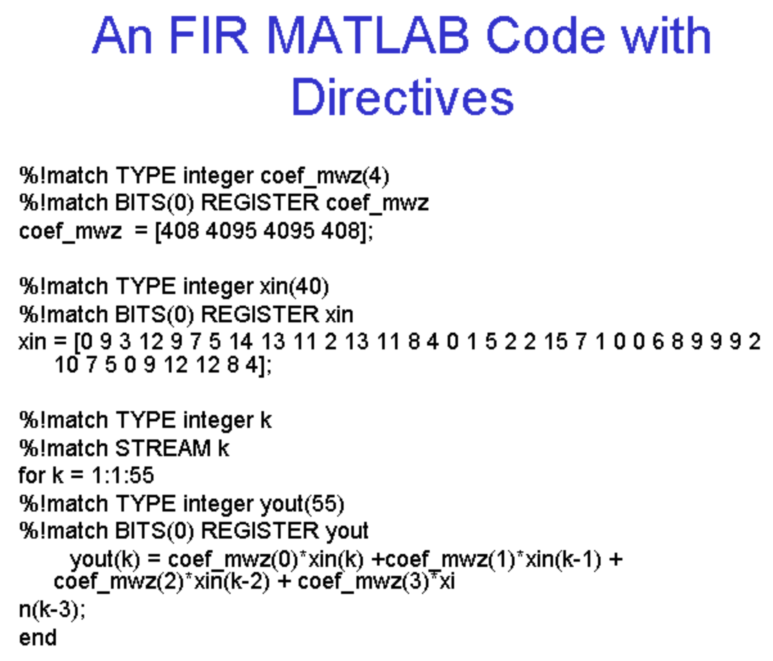


Figure 6. Use of the SYMPHANY automated tool on the various benchmarks.

Compiler Directives (Task 4): We have developed a complete set of directives to specify type, shape, size, precision, data distribution and alignment, task mapping, resource and timing constraints. The compiler recognizes many of these directives. Examples of such directives are:

```
%!MATCH SHAPE a(100,00)
%!MATCH TARGET WILDCHILD
%!MATCH STREAM
```

Figure 7 shows an example of a FIR filter code in MATLAB with directives.



An FIR MATLAB Code with Directives

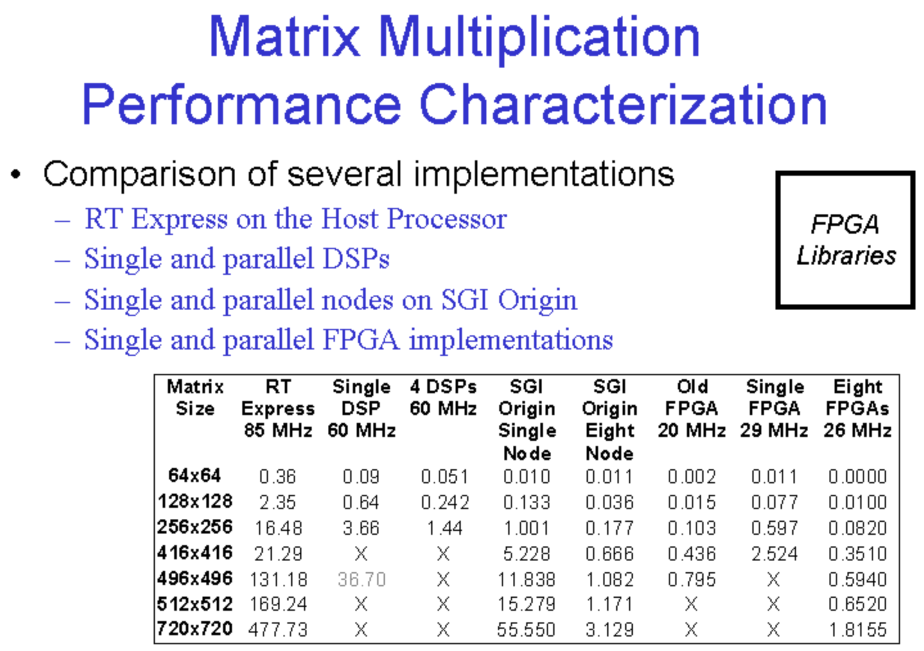
```
%!match TYPE integer coef_mwz(4)
%!match BITS(0) REGISTER coef_mwz
coef_mwz  = [408 4095 4095 408];

%!match TYPE integer xin(40)
%!match BITS(0) REGISTER xin
xin = [0 9 3 12 9 7 5 14 13 11 2 13 11 8 4 0 1 5 2 2 15 7 1 0 0 6 8 9 9 9 2
    10 7 5 0 9 12 12 8 4];

%!match TYPE integer k
%!match STREAM k
for k = 1:1:55
%!match TYPE integer yout(55)
%!match BITS(0) REGISTER yout
      yout(k) = coef_mwz(0)*xin(k) +coef_mwz(1)*xin(k-1) +
    coef_mwz(2)*xin(k-2) + coef_mwz(3)*xi
n(k-3);
end
```

Figure 7. An Example FIR code in MATLAB with Directives.

Applications (Task 5): We have looked at various adaptive applications in order to identify which libraries to implement. The applications include the FIR Filter, Matrix Multiplication, Sobel Transform, Average Filter, FFT, STAP application and the MPEG decoder, NASA Hyperspectral algorithms.

Libraries (Task 6): We have developed of various MATLAB libraries on the different platforms. The approach used was to develop each function as a parameterized function with the size of the data, the number of processors or FPGAs used, and the precision of the data (8 bit, 16 bit, 32 bit) for fixed point and floating point representations on three platforms. The platforms are the Annapolis Wildchild FPGA board, the Transtech DSP board and the Motorola embedded processor board. We have completed the development of the following library functions on the Wildchild FPGA board (using RTL VHDL and the commercial synthesis tools, namely Synplicity and Xilinx XACT place and route tools). (1) Real matrix addition (2) Real matrix multiplication (3) IIR and FIR Filtering (4) One and two-dimensional FFT. We have also developed the following library functions on the Transtech DSP board and the Motorola embedded board (using C plus MPI and the native C compilers for the Transtech and the Motorola boards). (1) Real and complex matrix addition (2) Real and complex matrix multiplication (3) One and two dimensional FFT. Each of these libraries has been developed with a variety of data distributions such as blocked, cyclic and block-cyclic distributions. We have characterized the performance of each of these library functions on various platforms for various data sizes and precision. In each case we have developed C program interfaces to our MATCH compiler so that the programs can be controlled from the host controller (Force board).

Some example results of the matrix multiplication library function on various targets are shown in Figure 8.



Matrix Multiplication Performance Characterization

- Comparison of several implementations
  - RT Express on the Host Processor
  - Single and parallel DSPs
  - Single and parallel nodes on SGI Origin
  - Single and parallel FPGA implementations

FPGA Libraries

Matrix Size	RT Express 85 MHz	Single DSP 60 MHz	4 DSPs 60 MHz	SGI Origin Single Node	SGI Origin Eight Node	Old FPGA 20 MHz	Single FPGA 29 MHz	Eight FPGAs 26 MHz
64x64	0.36	0.09	0.051	0.010	0.011	0.002	0.011	0.0000
128x128	2.35	0.64	0.242	0.133	0.036	0.015	0.077	0.0100
256x256	16.48	3.66	1.44	1.001	0.177	0.103	0.597	0.0820
416x416	21.29	X	X	5.228	0.666	0.436	2.524	0.3510
496x496	131.18	36.70	X	11.838	1.082	0.795	X	0.5940
512x512	169.24	X	X	15.279	1.171	X	X	0.6520
720x720	477.73	X	X	55.550	3.129	X	X	1.8155

Figure 8. Some example results of the matrix multiplication library function.

4. CONCLUSIONS

In conclusion we have developed the MATCH compiler which is capable of generating highly optimized hardware from applications described in MATLAB. A set of effective optimizations implemented in the compiler ensures that the quality of the output hardware is comparable to manually optimized hardware. The optimizations include parallelization, precision inferencing, IP core integration and pipelining. The effectiveness of the compiler was demonstrated by synthesizing hardware for a couple of signal/image processing applications. The outputs of the synthesized hardware were functionally verified against the outputs of the MATLAB interpreter. The execution times were almost equivalent to manual designed hardware, in fact superior in some cases were large amount of parallelism was available across loops. The resource utilizations were within a factor of four of the manual designs. All this was achieved while reducing the design time from months to minutes.

In terms of publications and patents, the MATCH project has:

- Supported 12 graduate students
- Produced 22 conference papers
- 3 Journal papers
- 10 Technical reports
- 4 Ph.D. theses
- 10 M.S. theses
- Three patents have been filed

Finally the technology has been transferred to a startup company called AccelChip, Inc. which has developed a commercial version of a tool called AccelFPGA and is selling it to various DSP customers.

5. PUBLICATIONS

- D.Chakrabarti and P. Banerjee, “Static Single Assignment Form for Message-Passing Programs,” *International Journal of Parallel Programming*, to appear, 2001.

- A.Nayak, M. Haldar, C. Chen, M. Sarrafzadeh, and P. Banerjee, “Power Optimizations in Delay Constrained Circuits,” *VLSI Design Journal*, to appear, 2001.

- M. Kandemir, P. Banerjee, A. Choudhary, J. Ramanujam, and E. Ayguade, “Static and Dynamic Locality Optimizations Using Integer Linear Programming,” *IEEE Transactions on Parallel and Distributed Systems*, to appear, 2001.

- Y. Yuan and P. Banerjee, “A Parallel Implementation of a Fast Multipole Based 3-D Capacitance Extraction Program on Distributed Memory Multicomputers,” *Journal of Parallel and Distributed Computing*, to appear, 2001.

- V. Kim, P. Banerjee, K. De, and J. Brouwers, “Parallel and Distributed VLSI Synthesis on a Network of Workstations,” *International Journal of Parallel and Distributed Systems and Networks*, to appear, 2001.

- V. Krishnaswamy, G. Hasteer, and P. Banerjee, “Automated Parallelization of Compiled Event Driven VHDL Simulation,” *IEEE Transactions on Computers*, to appear, 2001.

- D. Chakrabarti, “Global Optimization in Distributed Memory Message Passing Programs,” Ph.D. dissertation, Northwestern University, Aug. 2000.

- Y. Yuan, “Parallel Algorithms for 3D Extraction,” Ph.D. dissertation, Northwestern University, Aug. 2000.

- M. Haldar, “Optimized Hardware Synthesis for FPGAs,” Ph.D. dissertation, Northwestern University, Aug. 2001.

- A. Nayak, “Automatic Parallelization and Optimizations for Synthesizing MATLAB Programs on Multi-FPGA Systems,” Ph.D. dissertation, Northwestern University, Aug. 2001.

- S. Periyayacheri, A. Nayak, A. Jones, N. Shenoy, A. Choudhary, and P. Banerjee, ``Library Functions in Reconfigurable Hardware for Matrix and Signal Processing Operations in MATLAB," Proc. 11th IASTED Parallel and Distributed Computing and Systems Conference (PDCS'99), Cambridge, MA, Nov. 1999.

- M. Kandemir, A. Choudhary, J. Ramanujam, and P. Banerjee, ``On Reducing False Sharing While Improving Locality on Shared Memory Multiprocessors," Proc. 1999 International Conference on Parallel Architectures and Compilation Techniques (PACT'99)}, Newport Beach, CA, Oct. 12-16, 1999.

- Z. Ye, N. Shenoy, and P. Banerjee, ``A C Compiler for a Processor with a Reconfigurable Functional Unit," Proc. ACM/SIGDA Symposium on Field Programmable Gate Arrays, Monterey, CA, Feb. 2000.

- N. Shenoy, A. Choudhary, and P. Banerjee, ``A System-Level Synthesis Algorithm with Guaranteed Solution Quality," Proc. Design Automation and Test in Europe (DATE 2000)}, Paris, FRANCE, March 27-30, 2000.

- M. Haldar, A. Nayak, A. Choudhary, and P. Banerjee, ``Parallel Algorithms for FPGA Placement," Proc. Great Lakes Symposium on VLSI (GVLSI 2000), Evanston, IL, March 2000.

- P. Banerjee, N. Shenoy, A. Choudhary, S. Hauck, C. Bachmann, M. Haldar, P. Joisha, A. Jones, A. Kanhare, A. Nayak, S. Periyacheri, and M. Walkden, ``MATCH: A MATLAB Compiler for Distributed, Reconfigurable Computing Systems," submitted in Dec. 1999 to *FPGA Conference on Custom Computing Machines (FCCM2000),* Apr. 2000.

- Y. Yuan and P. Banerjee, ``A Parallel Implementation of A Fast Multipole Based 3-D Capacitance Extraction Program on Distributed Memory Multicomputers," *Proc. 14th International Parallel and Distributed Processing Symposium (IPDPS 2000),* Cancun, MEXICO, May 1-5, 2000 (Best Paper Award).

- Z. Ye, P. Banerjee, S. Hauck, and A. Moshovos, ``CHIMAERA: A High-Performance Architecture with a Tightly-Coupled Reconfigurable Functional Unit," *Proc. 27th International Symposium on Computer Architecture*, Vancouver, CANADA, June 10-14, 2000.

- M. Haldar, A. Nayak, A. Choudhary, P. Banerjee, "MATCH Virtual Machine: An Adaptive Runtime System to Execute MATLAB in Parallel," Proc. Int. Conf. Parallel Processing, Aug. 2000.

- Nayak, M. Haldar, A. Choudhary, P. Banerjee, "A Library Based Compiler to Execute MATLAB Programs on Heterogeneous Platforms," Proc. Parallel and Dist. Computing Systems (PDCS2000), Nov. 2000.

- V. Kim, P. Banerjee, K. De, and J. Brouwers, ``Parallel and Distributed VLSI Synthesis for Commericial CAD on a Network of Workstations," *Proc. 12th IASTED International Conference on Parallel and Distributed Computing Systems (PDCS 2000),* Las Vegas, NV, November 6-9, 2000.

- M. Haldar, A. Nayak, N. Shenoy, A. Choudhary, and P. Banerjee, ``FPGA Hardware Synthesis from MATLAB," Proc. of VLSI Design Conf. Jan. 2001, Bangalore, India.

- N. Shenoy, P. Banerjee, A. Choudhary, and M. Kandemir, ``Efficient Synthesis of Array Intensive Computations onto FPGA Based Accelerators," Proc. of VLSI Design Conf. Jan. 2001, Bangalore, India.

- M. Haldar, A. Nayak, A. Choudhary, and P. Banerjee, ``Automated Synthesis of Pipelined Designs on FPGAs for Signal and Image Processing Applications Described in MATLAB," Proc. Asia Pacific DAC, Mar. 2001.

- M. Haldar, A. Nayak, A. Choudhary, and P. Banerjee, ``"FPGA Hardware Synthesis from MATLAB Utilizing Optimized IP Cores" Proc. Ninth ACM/SIGDA International Symposium on Field Programmable Gate Arrays., Feb. 2001, San Jose, CA.

- A.Nayak, M. Haldar, A. Choudhary, P. Banerjee, ``Precision And Error Analysis Of MATLAB Applications During Automated Hardware Synthesis for FPGAs," Proc. Design Automation and Test in Europe (DATE 2001), Mar. 2001, Paris, France.

- M. Haldar, A. Nayak, A. Choudhary, P. Banerjee, ``A System for Synthesizing Optimized FPGA Hardware from MATLAB," Proc. Int. Conf. Computer Aided Design, Nov. 2001.

- A.Nayak, M. Haldar, A. Choudhary and P. Banerjee, ``Parallelization of MATLAB Applications for a Multi-FPGA System," Proc. FPGA Symp. on Custom Computing Machines (FCCM-2001), Napa Valley, CA, Apr. 2001.

- P. Joisha and P. Banerjee, "Correctly Detecting Intrinsic Type Errors in Typeless Languages Such as MATLAB," Proc. Of the APL Conf., New Haven, CT, Jun. 2001, to appear.

- D. Chakrabarti and P. Banerjee, "Global Optimization Techniques for Automatic Parallelization of Hybrid Applications," Proc. Int. Conf. Supercomputing, Sorento, Italy, Jun. 2001, to appear.

- A. K. Jones and P. Banerjee, "Parallel Implementation of Matrix and Signal Processing Libraries on FPGAs," *Proc. IASTED Parallel and Distributed Computing Systems Conference (PDCS 2001)*, Anaheim, CA, August 2001.

- P. Banerjee, M. Haldar, A. Nayak, and A. Choudhary, "Overview of the MATCH Compiler for Compiling MATLAB Programs into Hardware," *Proc. NASA Earth Science Technology Conference*, Washington DC, August 2001.

- P. Banerjee, A. Choudhary, M. Haldar and A. Nayak, "Method and Apparatus for Automatically Generating Hardware from Algorithms Described in MATLAB," U.S. Patent filed, Jan. 26, 2001.

- P. Joisha, P. Banerjee, N. Shenoy, "Methods for Array Shape Inferencing for a Class of Functions in MATLAB," U.S. Patent filed Jan. 31, 2001.

- Nayak, M. Haldar, N. Shenoy, A. Choudhary, P. Banerjee, "Design System and Method to Compile MATLAB programs on Heterogeneous Platforms Comprising FPGAs and Embedded Processors," U.S. Patent filed March 19, 2001.